:08:30 OCA PAD INITIATION - PROJECT HEADER INFORMATION 03/13/91

Active

roject #: E-21-691 Cost share #: Rev #: 0
enter # : 10/24-6-R7155-0A0 Center shr #: OCA file #:
Work type : RES
ontract#: LETTER DATED 13 FEBRUARY 1991 Mod #: Document : GRANT
rime #: Contract entity: GTRC

ubprojects ? : N
ain project #:

roject unit: ELEC ENGR Unit code: 02.010.118
roject director(s):
ALLEN, P E ELEC ENGR (404)894-6251

ponsor/division names: TEXAS INSTRUMENTS /
ponsor/division codes: 219 / 009

ward period: 910101 to 911231 (performance) 911231 (reports)

ponsor amount	New this change	Total to date
Contract value	15,000.00	15,000.00
Funded	15,000.00	15,000.00
ost sharing amount		0.00

oes subcontracting plan apply ?: N

itle: APPLICABILITY OF VHDL TO ANALOG IC DESIGN

PROJECT ADMINISTRATION DATA

A contact: E. Faith Gleason 894-4820

ponsor technical contact	Sponsor issuing office
R. SCOTT MARIN 214)480-1589	SAMUEL W. WEBSTER, JR., MGR. ED. AFF (214)917-7265
EXAS INSTRUMENTS, INC. 504 FORREST LANE AIL STOP 3143 ALLAS, TX 75266	TEXAS INSTRUMENTS, INC. P.O. BOX 650311 MAIL STOP 3986 DALLAS, TX 75265

curity class (U,C,S,TS) : U ONR resident rep. is ACO (Y/N): N
fense priority rating : supplemental sheet
uipment title vests with: Sponsor GIT
"EQUIPMENT PURCHASES ARE NOT CONSIDERED VALID COSTS AGAINST DSEG GRANTS.
ninistrative comments -
INITIATION.

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

Closeout Notice Date 02/10/92

oject No. E-21-691_____ Center No. 10/24-6-R7155-0A0_

oject Director ALLEN P E_____ School/Lab ELEC ENGR_____

onsor TEXAS INSTRUMENTS/_____

ntract/Grant No. LETTER DATED 13 FEBRUARY 1991_ Contract Entity GTRC

ime Contract No. _____

tle APPLICABILITY OF VHDL TO ANALOG IC DESIGN_____

fective Completion Date 911231 (Performance) 911231 (Reports)

---

oseout Actions Required:	Y/N	Date Submitted
Final Invoice or Copy of Final Invoice	Y	_____
Final Report of Inventions and/or Subcontracts	Y	_____
Government Property Inventory & Related Certificate	N	_____
Classified Material Certificate	N	_____
Release and Assignment	N	_____
Other _____	N	_____

Comments_____

_____

bproject Under Main Project No. _____

ntinues Project No. _____

stribution Required:

Project Director	Y
Administrative Network Representative	Y
GTRI Accounting/Grants and Contracts	Y
Procurement/Supply Services	Y
Research Property Managment	Y
Research Security Services	N
Reports Coordinator (OCA)	Y
GTRC	Y
Project File	Y
Other _____	N
_____	N

E: Final Patent Questionnaire sent to PDPI.

Six Months Progress Report

on

1991

RESEARCH ON APPLICABILITY OF VHDL TO ANALOG CIRCUITS

Submitted to:

Dr. Scott Marin

Texas Instruments, Inc.

P.O. Box 660246

Mail Stop 3143

Dallas, TX 75266

on

July 1991

by

Phillip E. Allen

School of Electrical Engineering

Georgia Institute of Technology

Atlanta, GA 30332

Introduction

This report summarizes the progress and plans related to the research performed in investigating applicability of VHDL to analog circuits at the School of Electrical engineering at Georgia Institute of Technology during the first 6 months of 1991.

Objective

The primary objective of the research performed during the period covered by this report was the investigation of the possibility of using VHDL to describe analog integrated circuits. The research is divided into two stages: 1) familiarity with the VHDL language and obtaining a VHDL simulator for digital circuits, and 2) apply VHDL to describe an analog integrated circuit such as an op amp and evaluation the capability of VHDL to describe and characterize analog circuits.

Progress

1) We have investigated VHDL, and can write VHDL programs for digital circuits and have successfully simulated simple examples of digital circuits on VSIM (a VHDL simulator from the University of Pittsburgh). Because VSIM implements a very restricted subset of VHDL, it is not appropriate for describing analog circuits. According to our understanding of the IEEE Standard 1076 for VHDL at this time, it should have the capability of describing analog circuits in time domain.

2) Apparently VSIM is not appropriate for analog circuits due to its restricted features. We couldn't implement operations other than logical operations. Also the signal types are restricted to bits, boolean and bitvectors, with no custom or defined signal types available. These features should be available in a complete VHDL simulator. We have tried to to implement arithmetic operations, such as multiplication, on VSIM, but it is difficult. It is only possible on logic gate level and it is not appropriate for practical purposes. Before continuing our work, we must get an appropriate compiler and simulator to implement the full VHDL language.

3) By the use of proper VHDL compiler and simulator, like the one distributed by MCC, we could work with analog circuits description and simulation in time domain. Some modification may be necessary to properly describe analog circuits, especially with regards to signal types. But there will still be difficulties in describing analog circuits in frequency domain (such as complex transfer functions).

4) In the next three months the major efforts will be:
- Install the MCC version of VHDL on our workstation and get it working.

- Do several typical examples of VHDL application on analog circuits.

- From the results of the examples, evaluating the capability of VHDL to describe and characterize analog circuits.

Other people working on this area

- B.R.Stanisic and M.W.Brown at IBM.
- C.M.Kurker and J.Paulos at North Carolina State University. B.S.Colen and E.S.Cooley at Dartmouth College.
- Hal Carter at University of Cinncinati.
- There is a committee at IBM for extending standard VHDL to analog circuits. We have contacted them and have requested minutes of their first meeting.

References

1. David R. Coelho. THE VHDL HANDBOOK. Kluwer Academic, Boston 1989.

2. R.Lipsett, C.Schaefer and C.Ussery. VHDL:HARDWARE DESCRIPTION AND DESIGN. Kluwer Academic Publisher, Boston 1988

3. Peter J. Ashenden. THE VHDL COOKBOOK

4. THE VHDL INSTALLATION NOTES. University of Pittsburgh, 1990

5. B.R.Stanisic and M.W.Brown, VHDL modeling for analog-digital hardware designs. IEEE Int. Conf. Comp. Aided Des. ICCAD 89 Dig. Tech., Pap. 1989.

6. C.M.Kurker, J.J.Paulos, B.S.Cohen and E.S.Cooley, Development of an analog hardware description language. Proceedings of the Custom Integrated Circuits Conference. 1990.

7. R.E.Harr and A.G.Stanculescu, APPLICATIONS OF VHDL TO CIRCUIT DESIGN. Kluwer Academic Publishers, 1991.

GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING
ATLANTA, GEORGIA 30332-0250

TELEPHONE: (404) 894-6251
FAX: (404) 853-9171
E-MAIL: pallen@monique.adgrp.gatech.edu

1/3/92

Dr. Scott Marin
Linear Design Manager
Texas Instruments
8504 Forrest Lane
P.O. Box 660246
Mail Stop 3143
Dallas, TX 75266

Dear Scott:

Enclosed is a preliminary copy of the Final Report of our 1991 research activities sponsored by your group at Texas Instruments. You should receive an official copy of this report after it passes through the proper channels here at Tech.

I had Xudong Shi write the report in order to help him improve his capability to write in English. I have gone over the report several times and have tried to convert it to readable English. I hope this does not cause a problem.

I appreciate your support of our research program. I look forward to working with you in 1992. I am excited about our 1992 work and hope that it will be a benefit to both of us. Please feel free to offer suggestions or ask any questions on our efforts at any time.

Best wishes for the New Year.

Very truly yours,

Phillip E. Allen
Schlumberger Professor
of Electrical Engineering

Final Report

on

1991

APPLICABILITY OF VHDL TO ANALOG IC DESIGN

Submitted to:

Dr. Scott Marin

Texas Instruments, Inc.

P.O. Box 660246

Mail Stop 3143

Dallas, TX 75266

on

December 1991

by

Phillip E. Allen

School of Electrical Engineering

Georgia Institute of Technology

Atlanta, GA 30332

Introduction

This report summarizes the progress and results related to the research conducted in investigating the applicability of VHDL for analog IC design at the School of Electrical Engineering at Georgia Institute of Technology during 1991. The report includes three examples of the application of VHDL to describe and simulate analog circuits. These examples are; 1) a 2-bit flash analog-to-digital converter, 2) an op amp described by using RC network, and 3) an op amp described by using state variable equations. Some important aspects concerned with analog IC design are demonstrated by these examples.

Objective

The primary objective of the research performed during the period covered by this report is to investigate the applicability of VHDL to analog IC design in a digital environment. VHDL is digitally oriented, but has been developed to describe every level of hardware from initial specification to gate level implementation. It provides the possibility to describe analog integrated circuits. The research was focused on using VHDL for analog applications. This report is divided into three stages: 1) the familiarization with VHDL, 2) the application of VHDL to describe the analog circuits,and 3) the evaluation of the capability of VHDL to describe analog circuits. The objectives of this research have been accomplished through the application of VHDL to different examples of analog circuits.

Progress

General description of the method and application

Three examples of analog, mixed analog-to-digital circuits have been described and simulated. The MCC VHDL simulator has used in this research[11-13]. Since all VHDL simulators are digitally oriented, it is difficult to describe and simulate analog circuits on digital simulator. The method adopted here use a combination of VHDL extensions of program language characters and higher level behavioral modeling of analog circuits. The higher level behavioral model provides the circuit input output relationship which is described by VHDL and simulated on the MCC simulator. The examples are representative of analog circuits. The method demonstrated in this research may be generalized to the VHDL description of analog circuits as shown in Fig-1. The linear, nonlinear and frequency characters of op amp were described in this manner.

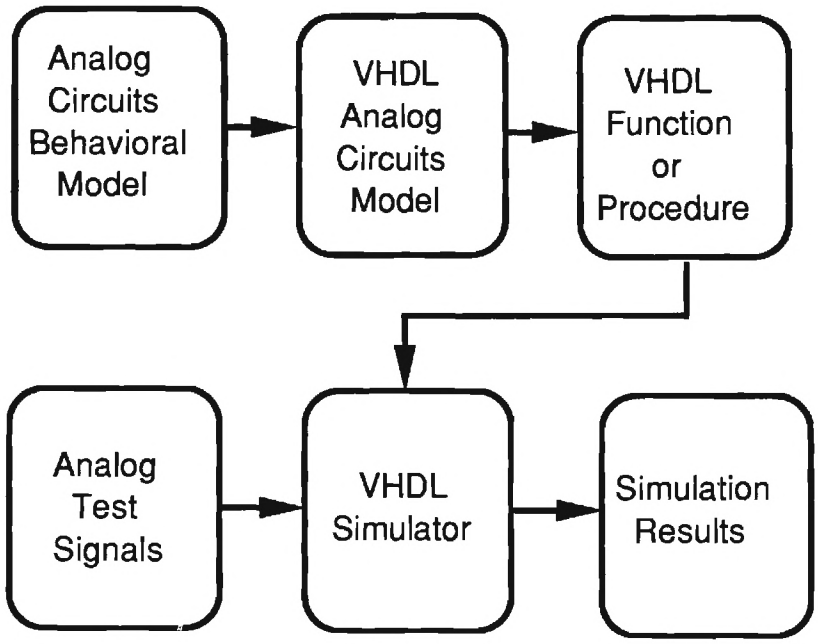

Fig- 1 VHDL description of analog circuits.

Examples

Example 1. 2-bit flash analog-to-digital convertor

The circuit shown in Fig-2 consists of three comparators, four resistors and eight digital gates.

Modeling Method

The relation of the amplitude of the comparators to the A/D converter output is given below:

$$Q0 = D1*\overline{D2} + D3$$

$$Q1 = D2 + D3$$

The comparators are modeled as ideal comparators and a delay of 20 ms.

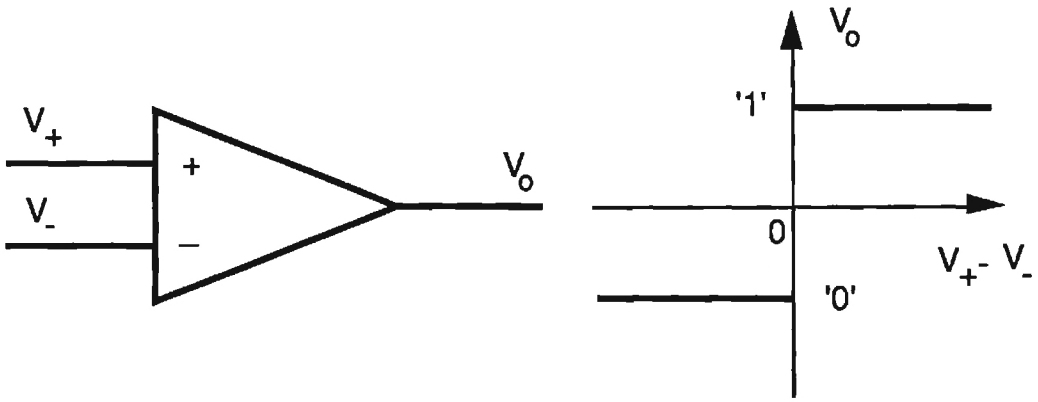

Fig-2 Ideal comparator.

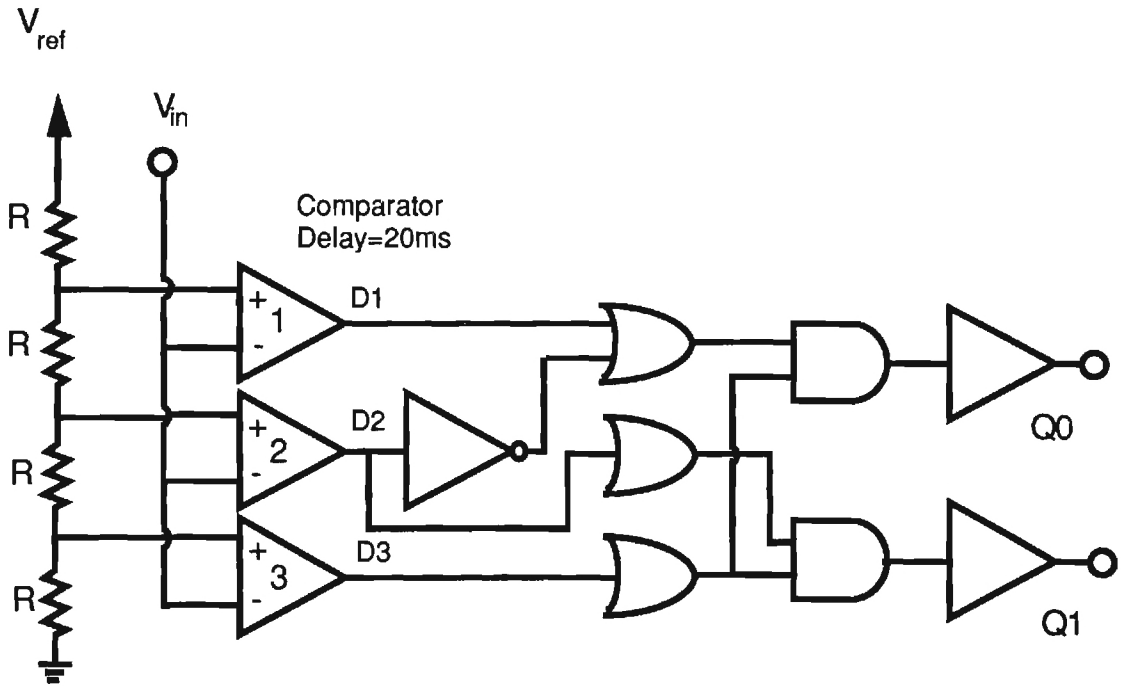

Fig - 3 Example of 2-bit flash A/D converter.

The VHDL function describing the comparator is given in Fig-4:

```
Function comparator( vp  :  real;
                     vin :  real ) return BIT is
variable vout : BIT := '0';
begin
        if( vin >= vp ) then
                vout := '1';
        else
                vout := '0';
        end if;
return vout;
end comparator;
```

Fig-4 Example of VHDL description for comparator

The comparator input is vp, an analog signal. The comparator output is a digital signal, '1' or '0'. The VHDL description "function comparator" accepts an analog signal and outputs a digital signal which is passed to digital gates. Fig-5 shows the time domain results of applying a ramp input to the 2-bit A/D converter of Fig-3.

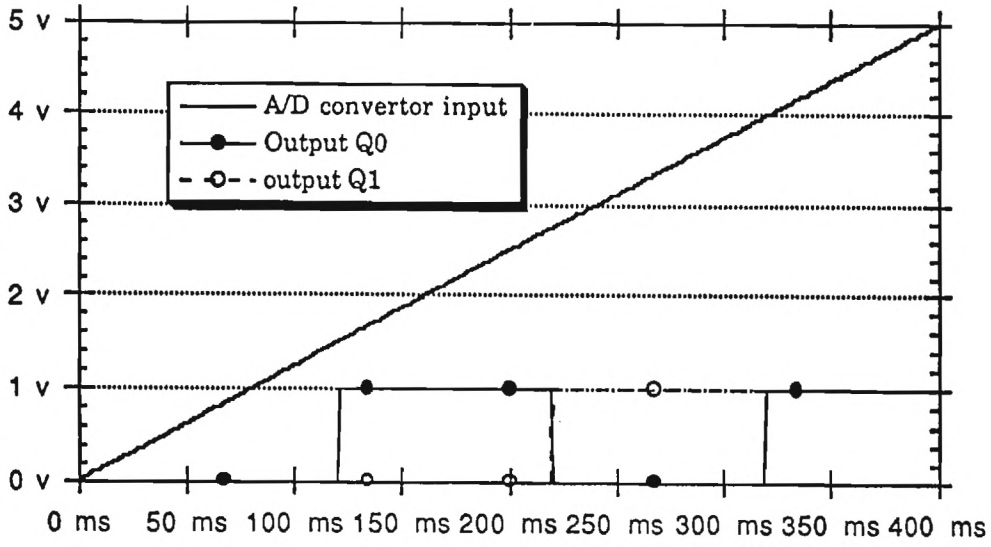

Fig-5 VHDL simulation result of 2-bit flash A/D converter

The VHDL function describing the comparator is given in Fig-4:

```
Function comparator( vp  :  real;
                     vin :  real ) return BIT is
variable vout : BIT := '0';
begin
        if( vin >= vp ) then
                vout := '1';
        else
                vout := '0';
        end if;
return vout;
end comparator;
```

Fig-4 Example of VHDL description for comparator

The comparator input is vp, an analog signal. The comparator output is a digital signal, '1' or '0'. The VHDL description "function comparator" accepts an analog signal and outputs a digital signal which is passed to digital gates. Fig-5 shows the time domain results of applying a ramp input to the 2-bit A/D converter of Fig-3.

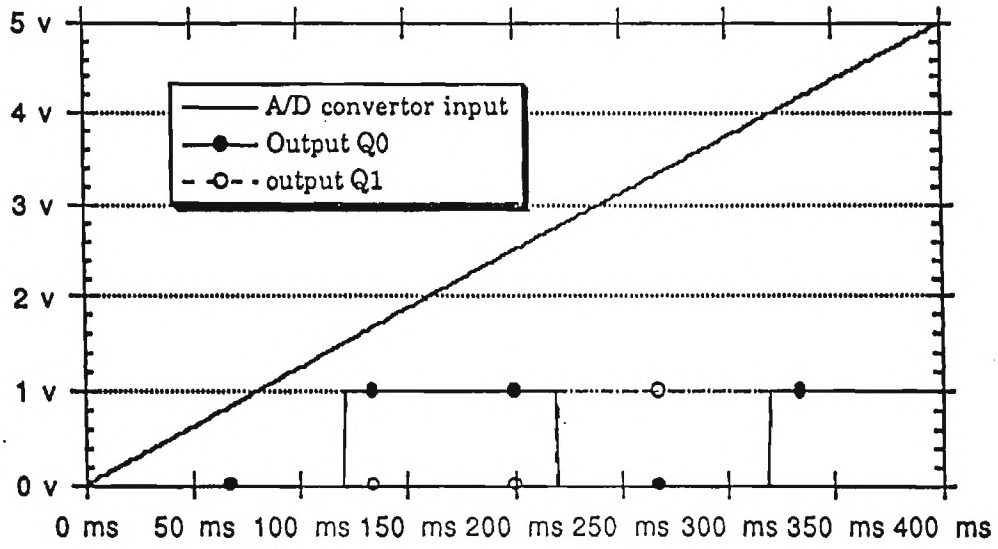

Fig-5 VHDL simulation result of 2-bit flash A/D converter

Example 2. Op Amp Time Domain Character

This example shows a method of using RC networks to model poles in op amp transfer function.

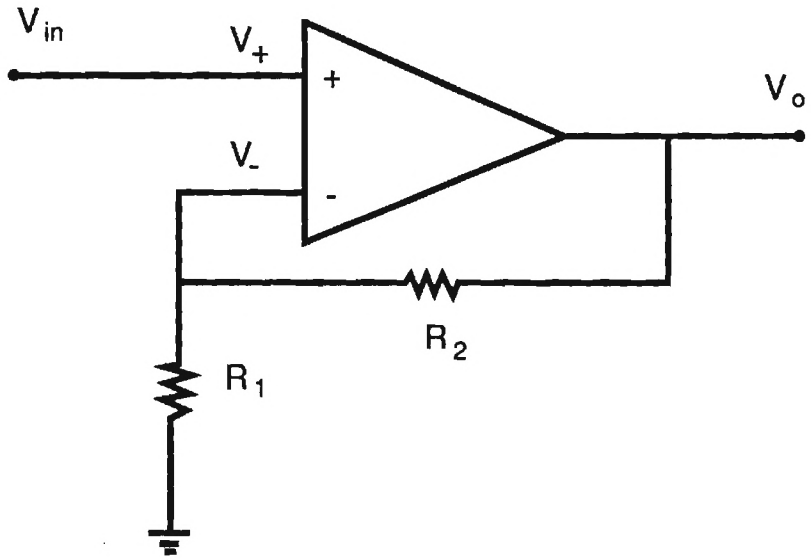

Fig - 6 Example of feedback op amp.

Modeling Method:

A single pole model for the op amp is given as

$$V_o = \frac{A_0}{ts+1} (V_+(s)-V_-(s))$$

R_i = finite

A_0 = amplitude of the differential gain of the op amp.

$R_o = 0$

t = op amp time constant.

Because of the instability encountered in feedback loop of a event driven simulator, it is necessary to write the closed-loop form of the transfer function in order to avoid feedback paths. The closed loop expression for this example is

$$\frac{V_o(s)}{V_+(s)} = \frac{(R_i\text{-}R)R_2}{R_iR\frac{R_2ts}{RA_0} + 1}$$

where

$$R = \frac{R_1R_2R_i}{R_1R_2+R_1R_i+R_2R_i}$$

The parameters in this example are $A_0=10^5$, R_i=1MW, R_1=1KW, R_2=9KW, t=1ms.

In the closed loop transfer function $\frac{1}{R_iR\frac{ts}{RA_0}}$ is equivalent to an RC network as shown in Fig-7.

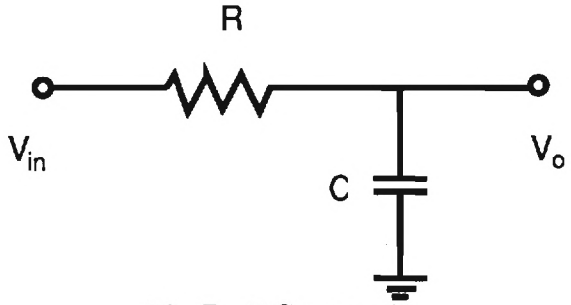

Fig-7 RC network.

The complex frequency domain transfer function of this network is

$$\frac{V_o(s)}{V_{in}(s)} = \frac{1}{RCs+1} \qquad 2\text{-}1)$$

Comparing this result with the above equation gives

$$R_iR\frac{t}{RA_0} = RC. \qquad (2\text{-}2)$$

In the time domain Eq. (2-1) becomes

$$V_{in} = RC\frac{dV_o}{dt} + V_o \qquad (2\text{-}3)$$

Using discrete time notation, we can write Eq. (2-3) as

$$\frac{dV_o}{dt}\Big|_{NT=NT} = \frac{V_o(NT)\text{-}V_o(NT\text{-}T)}{T} \qquad (2\text{-}4)$$

The desired result in a form suitable for an event driven simulator is

$$V_o = \frac{TV_{in}(NT)+RCV_o(NT-T)}{T+RC} \qquad (2\text{-}5)$$

where T is the time step. $V_o(NT)$ and $V_o(NT-T)$ are value of V_o at time NT and (NT-T), respectively.

Fig-8 gives a VHDL function which implements this discrete expression of RC network input-output relation.

```
Function rc ( r : real;
              c : real;
              prev_vout : analog_signal;
              vin : analog_signal )
              return analog_signal is;
```

Fig-8 VHDL description of RC network.

vin corresponds to V(NT) and prev_vout to V(NT-T). The analog_signal is a user defined signal by the use of VHDL signal defining capability.

The general form of transfer function having more than one pole can be developed in similar manner. Considering the following transfer function:

$$V_o(s) = \frac{A_0}{(\frac{s}{p_1}+1)(\frac{s}{p_2}+1)\ldots(\frac{s}{p_n}+1)} V_{in}(s) \qquad (2\text{-}6)$$

This form of transfer function is expressed in VHDL as following ideal op and cascade of rc network as shown in Fig-9.

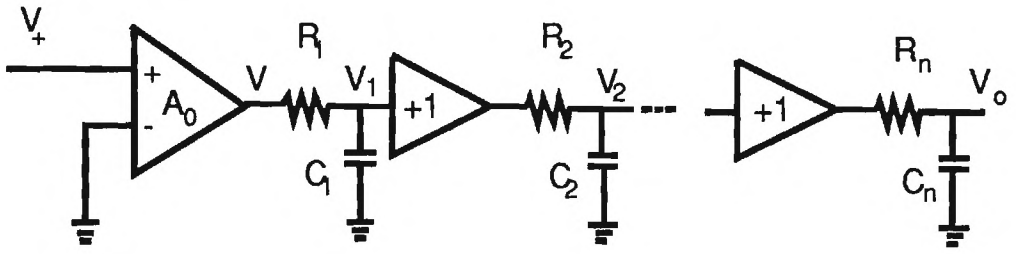

Fig-9 Cascade of RC network.

The ideal op output is V which is the input of first RC network stage. The first stage rc network output V_1 is the input of second RC network input and so on. Each RC network stage is implemented by the discrete form of Eq. (2-4) in VHDL. The VHDL simulation results compared with SPICE is shown in Fig-10.

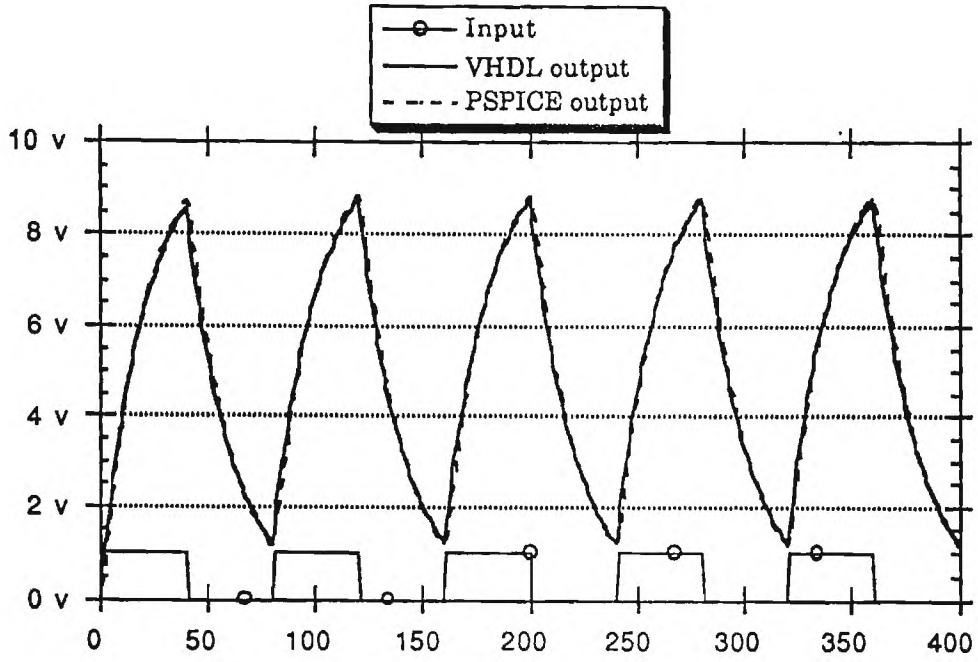

Fig-10 VHDL simulation result of op amp with one pole (time unit = 5ns)

Example 3 Op Amp Time Domain and Frequency Domain Characterization

Modeling Method

The open loop voltage gain of the op opm has the transfer function shown in form (3-1) which is comprised of differential mode transfer function, common mode transfer function, high voltage supply ripple transfer function, low voltage supply ripple transfer function and dc offset. We now want to describe its closed loop op amp time and frequency characteristics.

$$V_O = A_{vd}(s)(V_+(s)-V_-(s))+A_{vc}(s)(v_+(s)+V_-(s))+A_{vhh}(s)V_{hh}(s) +A_{vll}(s)V_{ll}(s)+V_{os} \quad (3\text{-}1)$$

Time Domain Characterization

Assume that the differential mode transfer function is given in in form of (3-2).

$$A_{vd}(s) = A_d\frac{(s+z_{11})(s+z_{12})\ldots(s+z_{m1})}{(s+p_{11})(s+p_{12})\ldots(s+p_{n1})} \quad (3\text{-}2)$$

The common mode transfer function is given in form of (3-3).

$$A_{vc}(s) = A_c\frac{(s+z_{21})(s+z_{22})\ldots(s+z_{m2})}{(s+p_{21})(s+p_{22})\ldots(s+p_{n2})} \quad (3\text{-}3)$$

Also assume that the high voltage supply ripple transfer function is given in in form of (3-4).

$$A_{vhh}(s) = A_h\frac{(s+z_{31})(s+z_{32})\ldots(s+z_{m3})}{(s+p_{31})(s+p_{32})\ldots(s+p_{n3})} \quad (3\text{-}4)$$

The low voltage supply ripple transfer function is given in form of (3-5).

$$A_{vll}(s) = A_l\frac{(s+z_{41})(s+z_{42})\ldots(s+z_{m4})}{(s+p_{41})(s+p_{42})\ldots(s+p_{n4})} \quad (3\text{-}5)$$

With feedback, the op amp has the closed loop frequency domain relationship in the general form of (3-6)

$$V_o(s)(a_0s^m+a_1s^{m-1}+\ldots+a_m) = V_+(s)(b_0s^n+b_1s^{n-1}+\ldots+b_n) \quad (3\text{-}6)$$

The equation (3-7) is the time domain counterpart of (3-6).

$$a_0v_o^{(m)}(t)+a_1v_o^{(m-1)}(t)+\ldots+a_mv_o(t) = b_0v_+^{(n)}\ldots+b_nv_+(t) \quad (3\text{-}7)$$

Now we set:

$$x_1=v_o,\ x_2=v_o^{(n)},\ \ldots\ldots,\ x_m=v_o^{(m-1)} \quad (3\text{-}8)$$

Equations (3-9) and (3-10) is the equivalent form of (3-8).

$$x_1^{(1)}=x_2,\ x_2^{(1)}=x3,\ \ldots\ldots,\ x_{m-1}^{(1)}=x_m \quad (3\text{-}9)$$

$$x_m^{(1)}= v_o^{(m)}= -\frac{1}{a_0}(a_0v_o^{(m-1)}+\ldots+a_mv_0)+\frac{1}{a_0}(b0v_o^{(m-1)}+\ldots+a_mv_o)$$
$$+\frac{1}{a_0}(b_0v_+^{(n)}+\ldots+b_nv_+) \quad (3\text{-}10)$$

As a result, the equivalent state variables equation of (3-9) and (3-10) in matrix form can be written as:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ . \\ . \\ . \\ \dot{x}_m \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & . & . & 0 \\ 0 & 0 & 1 & . & . & 0 \\ . & & & . & & . \\ . & & & & . & . \\ . & & & & & 1 \\ -\frac{a_m}{a_0} & -\frac{a_{m-1}}{a_0} & . & . & . & -\frac{a_1}{a_0} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_m \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{b_0 v_+^{(n)}}{a_0} + \ldots + \frac{b_n}{a_0} \end{bmatrix}$$

or:

$$\dot{X} = AX + U \tag{3-12}$$

The discrete form of equation (12) is as follows:

$$[\, X_n - X_{n-1} \,] / T = AX_{n-1} + U_{n-1} \tag{3-13}$$

$$Xn = T\,[\, AXn\text{-}1 + Un\text{-}1 \,] + Xn\text{-}1 \tag{3-14}$$

Here T is the time step, Xn and Xn-1 are the values of vector X at time NT and (NT-T), respectively.

Through equations (3-1)-(3-14) we can describe and simulate the time domain characteristics of analog circuits using VHDL. An op amp which has one zero and three poles has been tested in this manner.

Frequency Domain Characterization

We can describe the frequency response given the transfer function. For example, the differential mode transfer function $A_{vd}(s)$ is give in (3-15).

$$A_{vd}(j\omega) = Re(A_{vd}(j\omega)) + Im(A_{vd}(j\omega)) \tag{3-15}$$

The magnitude and phase response are expressed in form of (3-16) and (3-17).

$$M^2(\omega) = Re^2(A(j\omega)) + Im^2(A_{vd}(j\omega)) \tag{3-16}$$

$$F(j\omega) = \arctan\left[\frac{Re(A_{vd}(j\omega))}{Im(A_{vd}(j\omega))}\right] \quad (3\text{-}17)$$

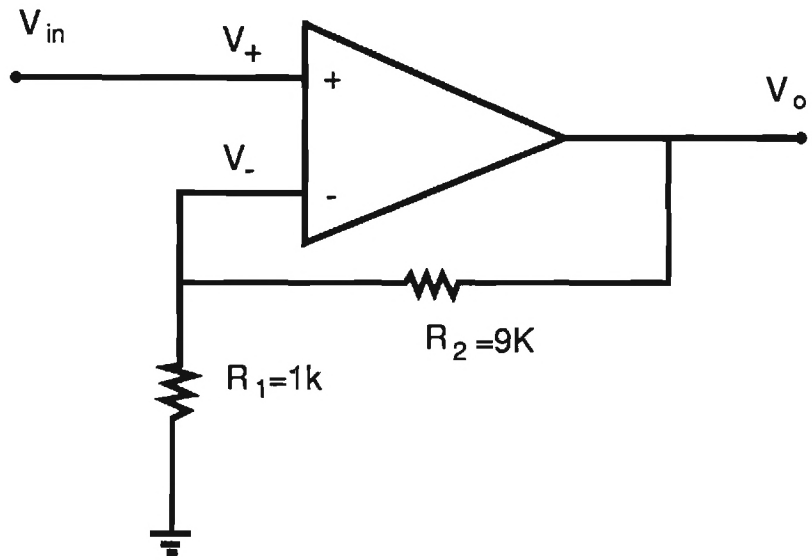

Fig - 11 Feedback op amp.

The open loop gain of the op amp in Fig-11 has the differential mode transfer function $A_{vd}(s)$:

$$V_o(s) = A_{vd}(s)(V_+(s)-V_-(s)) \quad (3\text{-}18)$$

where

$$A_{vd}(s) = A_d \frac{(1-\frac{s}{z_1})}{(1+\frac{s}{p_1})(1+\frac{s}{p_2})(1+\frac{s}{p_3})} \quad (3\text{-}19)$$

and $A_d=10^5$, $z_1=2x10^{-8}$, $p_1=10^{-2}$, $p_2=10^{-7}$, $p_3=10^{-8}$.

The closed loop transfer function is:

$$V_o(s) = \frac{A_d(R_1+R_2)R_i p_1 p_2 p_3 (z_1-s)/z_1}{rs^3+r(p_1+p_2+p_3)s^2+(r(p_1p_2+p_1p_3+p_2p_3)-A_dR_1R_ip_1p_2p_3/z_1)s+(r+A_dR_1R_i)p_1p_2p_3} \quad (3\text{-}20)$$

here

$$R_i = \text{input resistance of op} \quad (3\text{-}21)$$

and

$$r = \frac{R_1R_2R_i}{R_1R_2+R_1R_i+R_2R_i} \tag{3-22}$$

Time domain characterization can be modeled as shown in Fig-12. This approach uses a state variable equation using the form of Eq. (3-14) where,

$$a_0 = r \tag{3-23}$$

$$a_1 = r(p_1+p_2+p_3) \tag{3-24}$$

$$a_2 = r(p_1p_2+p_1p_3+p_2p_3)-A_dR_1R_ip_1p_2p_3/z_1 \tag{3-25}$$

$$a_3 = (r+A_dR_1R_i)p_1p_2p_3 \tag{3-26}$$

$$b_0 = -A_d(R_1+R_2)R_ip_1p_2p_3/z_1 \tag{3-27}$$

$$b_1 = A_d(R_1+R_2)R_ip_1p_2p_3 \tag{3-28}$$

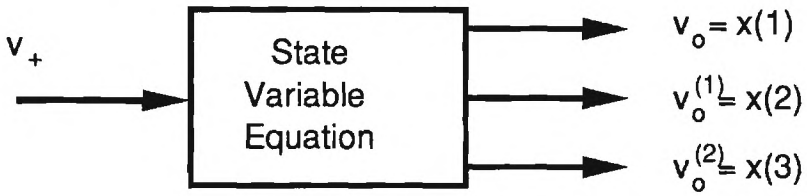

Fig-12 State Variable Equation.

The linear state variable equation model also allows us to implement nonlinear effects of the circuits. For example:

1.) For linear model , substitute (3-19)-(3-25) to equation (3-14) and use VHDL function to describe it.

2.) To model output voltage clamping, set $|x(1)| \leq$ voltage clamping.

3.) To model slew rate limitation, set $|x(2)| \leq$ slew rate limitation.

The results of VHDL time domain simulation compared with SPICE are shown in Fig-(13)-(15).

Frequency domain character is modeled as equations (3-16) and (3-17). The VHDL simulation results and corresponding SPICE results are shown in Fig-(16)-Fig(19).

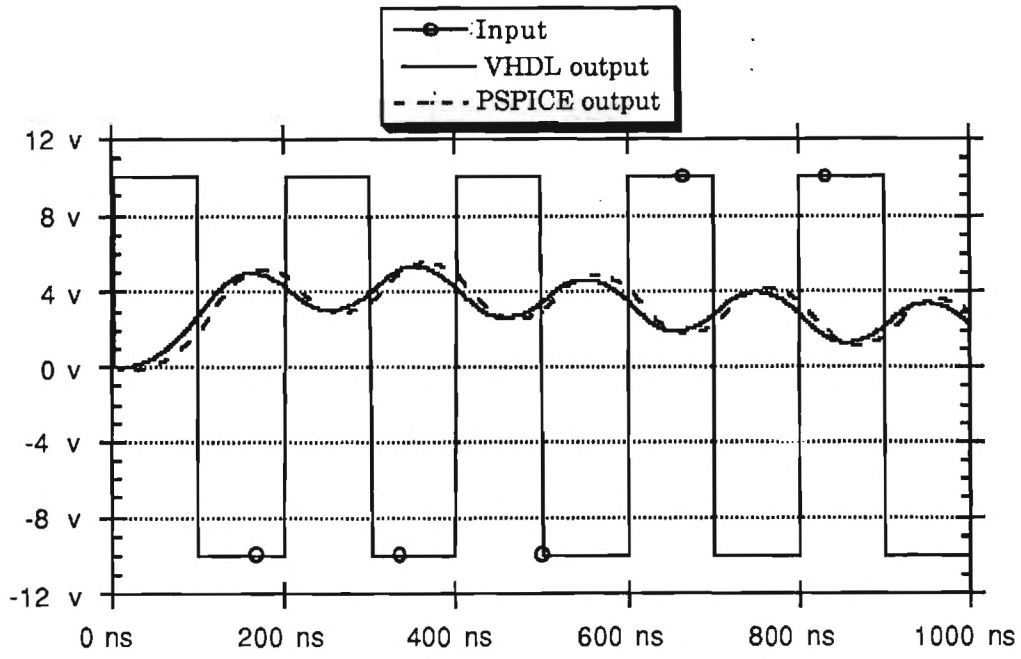

Fig-13 Simulation result for example 3 for the linear case.

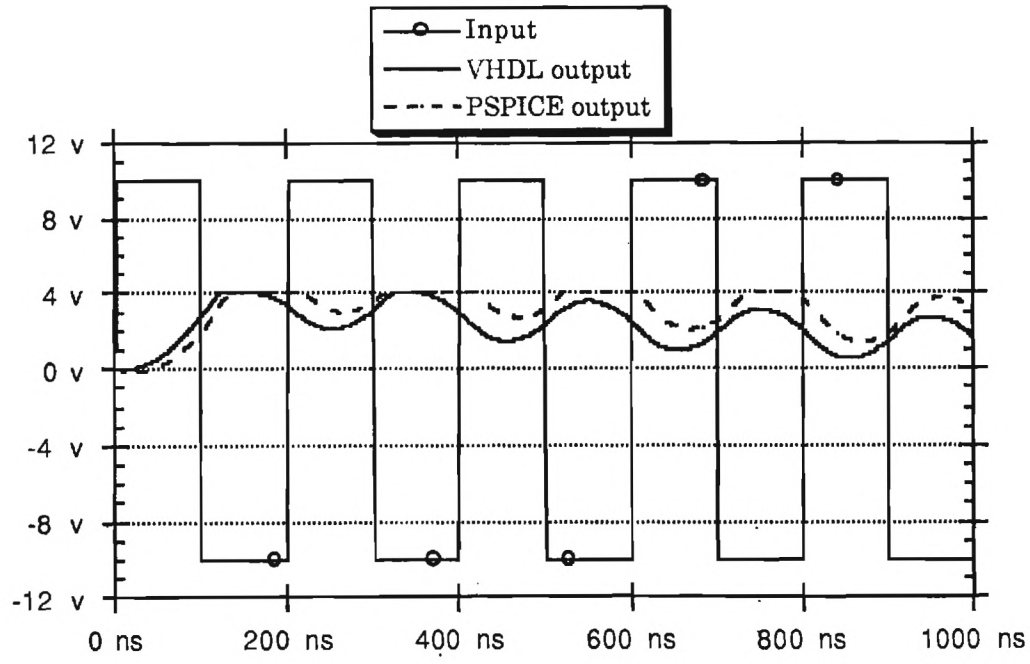

Fig-14 Simulation result for example 3 for theoutput clamping case.
(Output clamping = 4 volt)

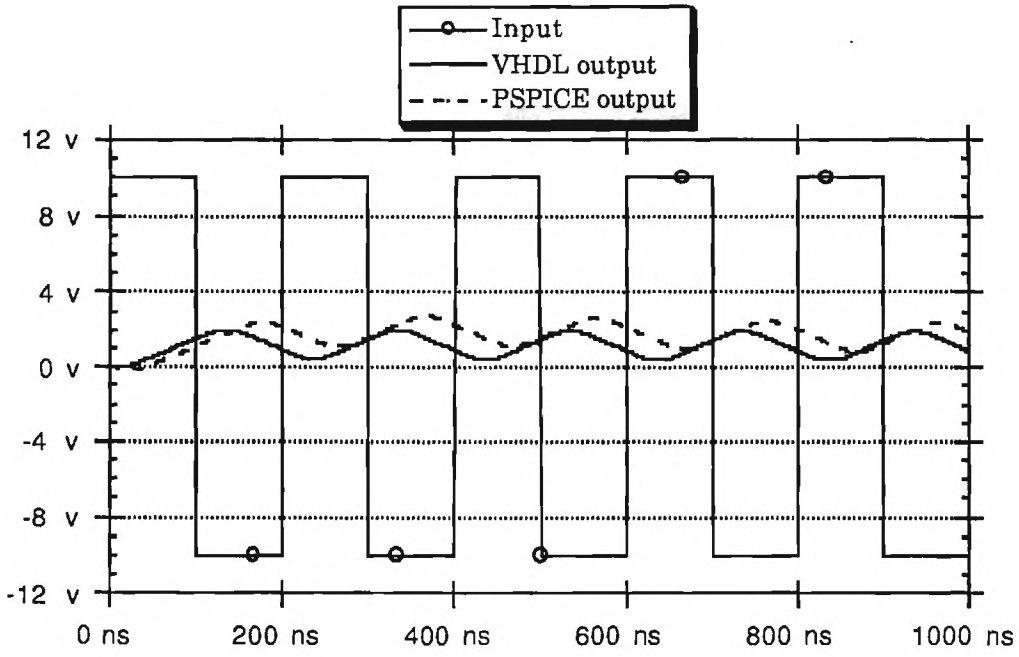

Fig-15 Simulation result for example 3 for the slew rate limitation case.
(Slew rate limit = 2×10^7)

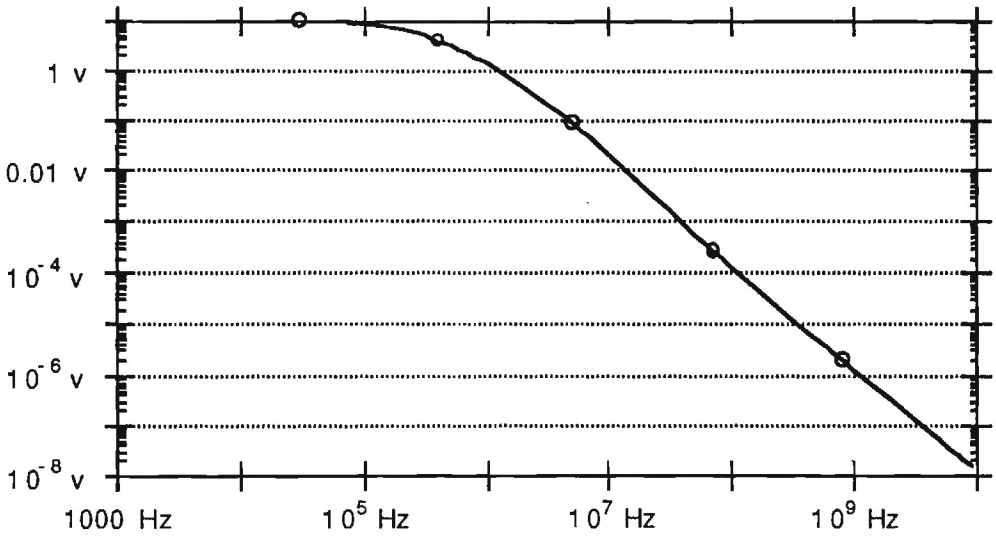

Fig-16 VHDL frequency magnitude response for example 3.

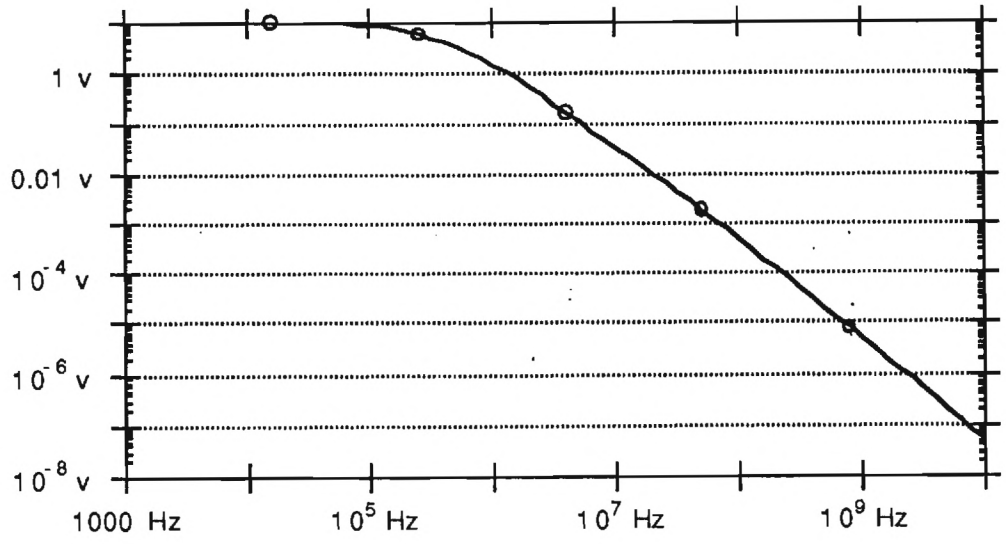

Fig- 17 SPICE frequency magnitude response for example 3.

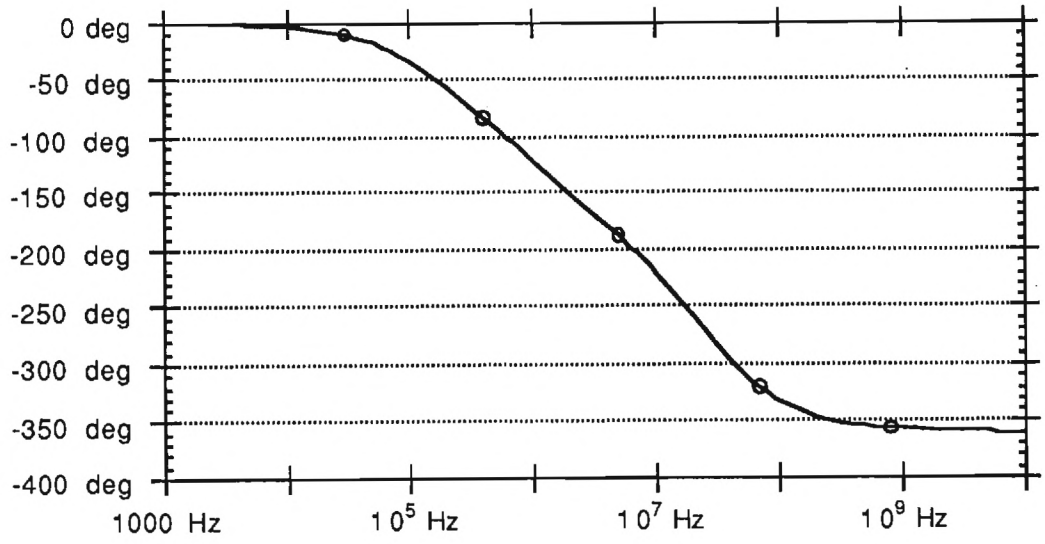

Fig-18 VHDL frequency phase response for example 3.

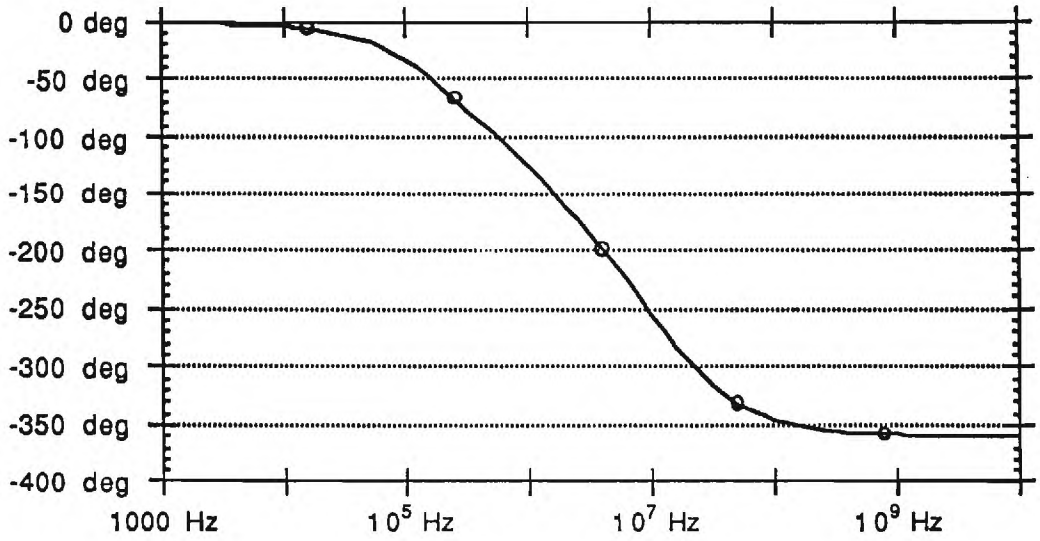

Fig-19 SPICE frequency phase response for example 3.

- **Results from examples:**

The examples show that the VHDL is capable of describing an analog-to-digital converter. For op amp the linear characteristics can be described. However in the nonlinear simulation there are some gaps between the VHDL results and SPICE results. These gaps are due to the method of modeling circuits. SPICE defined all electrical states in the circuits. By using state variable equation obtained from the transfer function, the linear model is equivalent to SPICE. However when trying to model nonlinear characteristics such as out voltage clamping and slew rate limitation discrepancies occur as shown in Fig-(9)-(11) .

While the output voltage in the state variable equation is equivalent to the SPICE output voltage, the higher order derivatives have no counterparts in SPICE model. Therefore, the nonlinearity of op amp specified in SPICE model is not equivalent to that in VHDL state variable equation. This problem may be solved by using appropriate modeling techniques in VHDL.

- **Limitations identified from examples:**

This research has identified several important limitations when using a VHDL simulator to model analog circuits. These limitations are:

1. There are no KCL/KVL in VHDL.

2. There are no analog signals available in VHDL. The user must generate them.
3. No standard arithmetic functions are provided by VHDL. It is necessary to generate analog signals and for frequency characterization. For example, the square root and arctan functions are essential for describing frequency response of the op amp.
4. Due to the discrete nature of VHDL simulator. The feedback loop can't be modeled on an open-loop basis. The time delay in feedback loop due to discretation of analog circuits may causes the unstable results for stable circuits.
5. VHDL simulator only has discrete time output because the simulator is event driven.

Work of Others

1. At University of Cincinnati Wen-Ying Zhou and Prof. H. Carter have implemented an AnaVHDL package. It is a SPICE algorithm implementation in VHDL. At present only R, C, L, V and I are implemented. The input takes SPICE input file. The output consists of nodal voltages. We have this package but haven't used it.
2. The VHDL Analog Sub-PAR(1076.1) Group is considering the necessity of modifying VHDL IEEE 1076 Standard to meet the requirements of analog circuits application. No modification has been made yet.

Conclusion

This research shows that the VHDL has the potential of describing the analog circuits in a digital environment. But in IEEE 1076 Standard this task is not easy because the existing simulators are all digitally oriented. This research has shown how to simulate linear analog circuits. While nonlinear characteristics, such as amplitude and slope limiting have been accomplished, a general approach to nonlinear analog circuits has not been developed. To appropriately describe nonlinearity of analog circuits the proper method of modeling for VHDL must be found.

Another problem is that VHDL is a hierarchical description language. The hierarchy in analog circuits is not as well defined as in digital circuits. Only behavior description of analog circuits has been described in VHDL, not the individual elements. The interface of digital parts and analog parts of circuits (A/D and D/A converters) can be described by behavior model in VHDL. Thus the major remaining work is how to accurately generate the necessary behavioral models.

Acknowledgement

During the period of this we have received the help from MCC, Prof. H. Carter at University of Cincinnati and Dr. Scott Marin at Texas Instrument, Inc., the graduate sudents Terry Sculley and Sub Choragudi at Georgia Institute of technology. We would like to express our gratitude for their help.

References

1. David R. Coelho. *The VHDL Handbook,* Kluwer Academic Publishers, Boston 1989.

2. R. Lipsett, C. Schaefer and C. Ussery. *VHDL: Hardware Description and Design,* Kluwer Academic Publishers, Boston, 1988.

3. Peter J. Ashenden. *The VHDL cookbook.*

4. B. R. Stanisic and M. W. Brown, "VHDL Modeling for Analog-digital Hardware Designs", IEEE Int. Conf. Comp. Aided. Des. ICCAD 89 Dig Tech Pap, 1989.

5. C. M. Kurker, J. J. Paulos, B. S. Cohen and E. S. Cooly, " Development of an Analog Hardware Description Language", Proceedings of the Custom Integrated Circuits Conference, 1990.

6. R. E. Harr and A. G. Stanculescu. *Applications of VHDL to Circuit Design,* Kluwer Academic Publisher, Boston,1991.

7. Wen-Ying Zhou and Harold Carter, "AnaVHDL: A Mixed-Mode Simulation Capability Using SPICE and VHDL", Proc. of 1991 Spring VHDL User's Group.

8. L. M. Augustin, D. C. Luckham, B. A. Gennart, Y. Huh and A. G. Stanculescu. *Hardware Design and Simulation in VAL/VHDL,* Kluwer Academic Publisher, Boston, 1991.

9. D. L. Perry. *VHDL,* Mcgraw-Hill, Inc, 1991.

10. "Minutes of VHDL Analog Sub-PAR group in San Francisco CA 6/21/91", Document Number: VIFANL-001.

11. "MCC VHDL System Version 3.2 General Release User's Guide", MCC Technical Report Number: CAD-83-91 (Q).

12. "MCC VHDL System Intermediate Representation Description", MCC Technical Report Number: CAD-086-91 Q.

13. "MCC VHDL System Version 3.2 General Release Magnetic Tape, Release Notes, and Installation Instruction", MCC Technical Report Number: CAD-103-91 (Q).

Attached Program of Examples to TI Final Report - 1991

1. VHDL program of example-1.
2. VHDL program of example-2.
3. VHDL program of example-3 (time domain)
4. VHDL program of example-3 (frequency domain)
5. VHDL program of package (define)
6. VHDL program of package (bus_res)
7. SPICE program of example-3 (time domain)
8. SPICE program of example-3 (frequency domain)

```
ample 1, 2-Bit Flash Analog to Digital Convertor

std.textio.all;
work.define.all;
ary MCC; Use MCC.c_Interface.all;
ty  adc is
  generic ( vref : real := 5.0 );
  port( vin : inout   real := 0.0 ;
        q0  : inout  bit  := '0' ;
        q1  : inout  bit  :='0');


adc ;


itecture beh  of adc is
al vp1 : real := 0.0;
al vp2 : real := 0.0;
al vp3 : real := 0.0;
al d0  : bit  := '0';
al d1  : bit  := '0';
al d2  : bit  := '0';
al d3  : bit  := '0';

tion comparator( vp   : real ;
                 vin  : real )
                return bit is
able vout : bit :='0';

   begin
        if( vin >= vp ) then
            vout:='1';
        else
            vout:='0';
        end if;
   return vout;
comparator;

n
  process
  variable simtime : real := 0.0;
    begin

    while ( now <= 401 ms  ) loop
        wait for 1 ms;
        simtime := simtime + 0.0125;
        vin <= simtime;
    end loop;
    c_procedure("exit",16);
  end process;
  vp1 <= vref*1.0/4.0;
  vp2 <= vref*2.0/4.0;
  vp3 <= vref*3.0/4.0;
  d0  <= '1' ;
  d1  <= transport comparator( vp1, vin ) after 20 ms;
  d2  <= transport comparator( vp2, vin ) after 20 ms;
  d3  <= transport comparator( vp3, vin ) after 20 ms;
  q0  <= d3 or ( d1 and ( not d2 ) );
  q1  <= d2 or d3;


itor : process ( vin, q0, q1 )
  file outfile : text is out "outfile";
  variable l : line;
```

```
variable a_v : real :=0.0;
variable b_v : bit  :='0';
variable c_v : bit  :='0';
variable a   : real :=0.0;
variable b   : bit  :='0';
variable c   : bit  :='0';
variable t   : time :=0 ms;
variable pt  : time :=0 ms;

    begin
       a := a_v;
       b := b_v;
       c := c_v;
       t := pt;
       a_v := vin;
       b_v := q0;
       c_v := q1;
       pt  := now/1000/1000;
       if(  pt > t ) then
       write( l, "time=" );
       write( l, t, field => 8 );
       write( l, "    vin= " );
       write( l, " ");
       write( l, a );
       write( l, "    q0= " );
       write(l, " " );
       write( l, b );
       write( l, "    q1= " );
       write( l, " " );
       write( l, c );
       writeline( outfile, l );
       else
     end if;
end process monitor;

beh;
```

```
ample 2, operational amplifer with 1-zero.

std.textio.all;
work.define.all;
ary MCC; Use MCC.C_Interface.all;
ty opm_ex2 is
 generic( gain : real := 100000.0;
          r    : real := 1000.0;          -- om
          c    : real := 0.000001;        -- F
          r1   : real := 1000.0;          -- om
          r2   : real := 9000.0;          -- om
          ri   : real := 1000000.0);      -- om


 port( vinn : inout  analog_signal:= (val =>0.0, kind =>v) ;
       vinp : inout  analog_signal:= (val =>0.0, kind =>v) ;
       vout : inout analog_signal := (val =>0.0, kind =>v) );


opm_ex2 ;

itecture beh  of opm_ex2 is
al vout_opm : real :=0.0;
al v1       : real :=0.0;
al pv1      : real :=0.0;
al pv2      : real :=0.0;
tant analog_time_delta : real :=5.0;          -- ns
tant analog_time_delay : time :=5 ns;

function for square wave

tion sq( x : real ) return real is
   variable sq_wave : real := 1.0;
   variable dev     : real := 0.0;
   variable n       : integer;
   variable p       : real := 0.0;
   constant c1      : real := 200.0;  --period of square pulse unit ns

   begin
        dev := x/c1;
        for i in 1 to 1000 loop
             p := P+1.0;
             if (  dev <= p and dev > p-1.0) then
                n :=i;
                exit;
             else
             end if;
        end loop;

        if ( n rem 2 = 1 ) then
            sq_wave := 1.0;
        else
            sq_wave := 0.0;
        end if;
rn sq_wave;
sq;

function for RC low pass

tion rc ( res : real;
          cap : real;
          gain : real;
          r1   : real;
          r2   : real;
```

```
        ri    : real;
        predl_vout : real;
        vin : real)
        return real is
   variable r    : real;
   variable tau  : real;
   variable vout : real;

   begin
        r     := r1*r2*ri/(r1*r2+r1*ri+r2*ri);
        tau   := (r2/r/gain)*res*cap;                    -- sec
        tau   := tau*1000.0*1000.0*1000.0;               -- ns
        vout  := (( predl_vout*tau ) +
                  vin*analog_time_delta) /
                  ( tau + analog_time_delta );
   return vout;
   end rc;

function for ideal opm

tion op(  gain : real;
          r1   : real;
          r2   : real;
          ri   : real;
          v    : real)
        return real is
able r      : real;
able a      : real;
able vout_1 : real;
    begin
         r      := r1*r2*ri/(r1*r2+r1*ri+r2*ri);
         a      := (ri-r)*r2/ri/r;
         vout_1 := v*a;
    return vout_1;
op;


n

ocess
riable simtime : real := 0.0;
 begin

    while ( now <= 2005 ns ) loop
      wait for analog_time_delay;
      simtime := simtime + analog_time_delta;
      vinp.val <= sq ( simtime );
      vinn.val <= -sq ( simtime );
    end loop;
    c_procedure("exit",16);
d process;
    vout_opm <= op( gain, r1, r2, ri, vinp.val);
    vout.val <= rc( r, c, gain, r1, r2, ri, pv1, vout_opm );
    pv1      <=  transport vout.val after analog_time_delay;

tor : process ( vinp, vout )
  file outfile : text is out "outfile";
  variable l : line;
  variable a_v : real :=0.0;
  variable b_v : real :=0.0;
  variable a   : real :=0.0;
  variable b   : real :=0.0;
  variable t   : time :=0 ms;
  variable pt  : time :=0 ms;
```

```
    begin
       a := a_v;
       b := b_v;
       t := pt;
       a_v := vinp.val;
       b_v := vout.val;
       pt  := now/5;
       if( pt > t ) then
       write( l, "time=" );
       write( l, t, field => 8 );
       write( l, "    vin= " );
       write( l, " ");
       write( l, a );
       write(l, " " );
       write( l, "    vout= " );
       write( l, " " );
       write( l, b );
       writeline( outfile, l );
       else
     end if;
end process  monitor;
beh;
```

```
--Example 3, operational amplifer time domain, closed loop, 1-zero, 3-poles.

use std.textio.all;
use work.define.all;
Library MCC; Use MCC.C_Interface.all;
entity opm_ex3 is
     generic( gain : real := 1.0E5;
              ri   : real := 1.0E6;        -- ohm
              r1   : real := 1.0E3;        -- ohm
              r2   : real := 9.0E3;        -- ohm
              z1   : real := 2.0E8;
              p1   : real := 1.0E2;
              p2   : real := 1.0E7;
              p3   : real := 1.0E8;
              v_clamp : real := 4.0;       -- volt
              slew_lim : real := 2.0E7);


     port( vinn : inout  analog_signal:= (val =>0.0, kind =>v) ;
           vinp : inout  analog_signal:= (val =>0.0, kind =>v) ;
           vout : inout analog_signal := (val =>0.0, kind =>v) );


end opm_ex3 ;

architecture beh  of opm_ex3 is
signal vpn : real :=0.0;
signal vpp : real :=0.0;
signal x_n      : vector :=(0.0,0.0,0.0);
signal x_n_1    : vector :=(0.0,0.0,0.0);
constant analog_time_delta : real :=1.0;          -- ns
constant analog_time_delay : time :=1 ns;

-- function for square wave

function sq( x : real ) return real is
       variable sq_wave : real := 1.0;
       variable dev     : real := 0.0;
       variable n       : integer;
       variable p       : real := 0.0;
       constant c1      : real := 100.0;  --period of square pulse unit ns

       begin
            dev := x/c1;
            for i in 1 to 1000 loop
                 p := P+1.0;
                 if (  dev <= p and dev > p-1.0) then
                    n :=i;
                    exit;
                 else
                 end if;
            end loop;
            if ( n rem 2 = 1 ) then
                sq_wave := 10.0;
            else
                sq_wave := -10.0;
            end if;
return sq_wave;
end sq;

-- function for state variable equation

function state(x : vector;
               gain  :  real;
```

```
                ri     :  real;
                r1     :  real;
                r2     :  real;
                z1     :  real;
                p1     :  real;
                p2     :  real;
                p3     :  real;
                vip_p  :  real;
                vin_p  :  real;
                vip : real;
                vin : real ) return vector is
variable  A : matrix;
variable  y : vector;
variable  b0, b1, a0, a1, a2, a3, v0, v1, r : real;
begin
          r  := r1*r2+r1*ri+r2*ri;
          b0 := -gain*p1*p2*p3*ri*(r1+r2)/z1;
          b1 := gain*p1*p2*p3*ri*(r1+r2);
          a0 := r;
          a1 := r*(P1+p2+p3);
          a2 := r*(p1*p2+p1*p3+p2*p3)-gain*p1*p2*p3*r1*ri/z1;
          a3 := p1*p2*p3*(r+gain*r1*ri);
          v0 := vip-vin;
          v1 :=(v0-(vip_p-vin_p))/(analog_time_delta/1000.0/1000.0/1000.0);
          for i in 1 to 3 loop
             y(i):=0.0;
             for j in 1 to 3 loop
                A(i,j):=0.0;
             end loop;
          end loop;
          A(1,2):=1.0;
          A(2,3):=1.0;
          A(3,3):=-a1/a0;
          A(3,2):=-a2/a0;
          A(3,1):=-a3/a0;
          for j in 1 to 3 loop
             for i in 1 to 3 loop
              y(j):=y(j)+analog_time_delta*A(j,i)*x(i)/1000.0/1000.0/1000.0;
             end loop;
          end loop;
           y(3):=y(3)+(b0*v1/a0+b1*v0/a0)*analog_time_delta/1000.0/1000.0/1000.0
                 +x(3);
           y(2):=y(2)+x(2);
           y(1):=y(1)+x(1);
           if( y(1)>v_clamp ) then
                y(1) := v_clamp;
           elsif( y(1)<-v_clamp ) then
                y(1) := -v_clamp;
           end if;
           if( y(2)>slew_lim ) then
                y(2) := slew_lim;
           elsif( y(2)<-slew_lim ) then
                y(2) := -slew_lim;
           end if;
          return y;
end state;

begin

stimulus:  process
  variable  simtime : real := 0.0;
  begin
        while ( now <= 1001 ns ) loop
          wait for analog_time_delay;
          simtime := simtime + analog_time_delta;
          vinp.val <= sq ( simtime );
```

```
        vinn.val <= 0.0;
      end loop;
      c_procedure("exit",16);
  end process stimulus;
        x_n <= state(x_n_1,gain,ri,r1,r2,z1,p1,p2,p3,vpp,vpn,
                     vinp.val, vinn.val);
        x_n_1 <= x_n after analog_time_delay;
        vpp <= vinp.val after analog_time_delay;
        vpn <= vinn.val after analog_time_delay;

monitor : process ( vinp, vinn, x_n(1) )
    file outfile : text is out "outfile";
    variable l : line;
    variable a_v : real :=0.0;
    variable b_v : real :=0.0;
    variable c_v : real :=0.0;
    variable a   : real :=0.0;
    variable b   : real :=0.0;
    variable c   : real :=0.0;
    variable t   : time :=0 ns;
    variable pt  : time :=0 ns;

    begin
          a := a_v;
          b := b_v;
          c := c_v;
          t := pt;
          a_v := vinp.val;
          b_v := vinn.val;
          c_v := x_n(1);
          pt  := now;
          if( pt > t ) then
          write( l, "time=" );
          write( l, t, field => 8 );
          write( l, " vin= " );
          write( l, " ");
          write( l, a );
          write( l, " vip= ");
          write( l, b );
          write(l, " " );
          write( l, " vout= " );
          write( l, " " );
          write( l, c );
          writeline( output, l );
          else
          end if;
   end process  monitor;
end beh;
```

```
ample 3, operational amplifer frequency domain.

std.textio.all;
work.define.all;
ary MCC; Use MCC.C_Interface.all;
ty opm_freq is
 generic( gain : real := 1.0E5;
          ri   : real := 1.0E6;        -- ohm
          r1   : real := 1.0E3;        -- ohm
          r2   : real := 9.0E3;        -- ohm
          z1   : real := 2.0E8;
          p1   : real := 1.0E2;
          p2   : real := 1.0E7;
          p3   : real := 1.0E8;
          fst  : real := 1.0E3;
          fend : real := 1.0E10);

 port( vinn : inout  analog_signal:= (val =>0.0, kind =>v) ;
       vinp : inout  analog_signal:= (val =>0.0, kind =>v) ;
       vout : inout analog_signal := (val =>0.0, kind =>v) );

opm_freq ;

itecture beh  of opm_freq is
al vpn : real :=0.0;
al vpp : real :=0.0;
al ss  : real;
al x_n      : vector :=(0.0,0.0,0.0);
al x_n_1    : vector :=(0.0,0.0,0.0);
tant analog_time_delta : real :=1.0;          -- ns
tant analog_time_delay : time :=1 ns;



nction for square root
tion sqrt( x : real ) return real is
able sum, x0 : real;
n
    sum := x/2.0;
    for i in 1 to 1000 loop
    x0 := sum;
    sum :=(x0+x/x0)/2.0;
    if( abs(sum-x0)/sum < 0.01 ) then
    return sum;
    end if;
    end loop;
sqrt;

unction for arctan

tion arctan( x : real ) return real is
able angle, p, temp : real;
n
    if( x = 1.0 ) then
            angle := 45.0;
            return angle;
    elsif( x = -1.0 ) then
            angle := -45.0;
            return angle;
    elsif( abs(x) < 1.0 ) then
            angle := 0.995354*x-0.288679*x*x*x+0.079331*x*x*x*x*x;
            angle := 180.0*angle/3.1415;
            return angle;
    elsif( x > 1.0 ) then
```

```
    angle := 3.1415/2.0-(0.995354/x-0.288679/x/x/x+0.079331/x/x/x/x/x);
            angle := 180.0*angle/3.1415;
            return angle;
    else
    angle := -3.1415/2.0-(0.995354/x-0.288679/x/x/x+0.079331/x/x/x/x/x);
            angle := 180.0*angle/3.1415;
            return angle;
    end if;
arctan;


unction for frequency response
tion freq ( f    :  real;
            gain :  real;
            ri   :  real;
            r1   :  real;
            r2   :  real;
            z1   :  real;
            p1   :  real;
            p2   :  real;
            p3   :  real) return vector is

able  y  :  vector;
able  re1,im1,re2,im2,m1,m2,m : real;
able  w , r, g,ang, ang1, ang2 :  real;
n
    r   := r1*r2+r1*ri+r2*ri;
    w   := 2.0*3.141516*f;
    g   := (r1+r2)*ri*gain*p1*p2*p3;
    re1 := 1.0;
    im1 := -w/z1;
    re2 := (-r*(p1+p2+p3)*w*w+p1*p2*p3*(r+r1*ri*gain))/g;
    im2 := (-r*w*w*w+(r*(p1*p2+p1*p3+p2*p3)-p1*p2*p3*r1*ri*gain/z1)*w)/g;
    m1  := re1*re1+im1*im1;
    m2  := re2*re2+im2*im2;
    m   := m1/m2;
    y(1):= sqrt( m );
    ang1 := arctan( im1 );
    if( abs(im2/1.0e15)>abs(re2) ) then
            if( im2>=0.0 ) then
                    ang2 := 90.0;
            else
                    ang2 := -90.0;
            end if;
    elsif( re2>0.0) then
            ang2 := arctan(im2/re2);
    elsif( im2>=0.0 ) then
            ang2 := arctan(im2/re2)+180.0;
    elsif( im2<0.0 ) then
            ang2 := arctan(im2/re2)+180.0;
    end if;
    y(2) := ang1-ang2;
n y;
freq;


ction for generating freq swap

ion swp( fst : real;
         fend: real;
         seed: real ) return real is
ble      f : real;


  f := seed;
```

```
 if( f<=fend and f >=fst ) then
   if ( f < 10.0 ) then
           f :=f+1.0;
   elsif( f < 100.0 ) then
           f :=f+10.0;
   elsif( f < 1.0e3 ) then
           f := f+1.0e2;
   elsif( f < 1.0e4 ) then
           f := f+1.0e3;
   elsif( f < 1.0e5 ) then
           f := f+1.0e4;
   elsif( f < 1.0e6 ) then
           f:=f+1.0e5;
   elsif( f < 1.0e7 ) then
           f := f+1.0e6;
   elsif( f < 1.0e8 ) then
           f := f+1.0e7;
   elsif( f < 1.0e9 ) then
           f := f+1.0e8;
   elsif( f <1.0e10 ) then
           f := f+1.0e9;
   end if;
   else
           f :=1.01*f;
   end if;
rn f;
swp;

ılus : process
able seed : real;
n
   vinn.val <= fst;
   while ( now <= 200 ns ) loop
     wait for 1 ns;
     seed := vinn.val;
     vinn.val <= swp( fst, fend ,seed );
   if vinn.val > fend then
           exit;
   end if;
   end loop;
   c_procedure("exit",16);
rocess stimulus;
       x_n <= freq(vinn.val,gain,ri,r1,r2,z1,p1,p2,p3);
       vpp <= arctan(-2.0);
or : process ( vinn.val, x_n(1),x_n(2))
file outfile : text is out "outfile";
variable l : line;
variable a_v : real :=0.0;
variable b_v : real :=0.0;
variable c_v : real :=0.0;
variable a   : real :=0.0;
variable b   : real :=0.0;
variable c   : real :=0.0;
variable t   : time :=0 ns;
variable pt  : time :=0 ns;
n
       a := a_v;
       b := b_v;
       c := c_v;
       t := pt;
       a_v := vinn.val;
       b_v := x_n(1);
       c_v := x_n(2);
       pt  := now;
       if( pt > t ) then
```

```
        write( l, " freq= " );
        write( l, " ");
        write( l, a );
        write( l, " mag= ");
        write( l, b );
        write(l, " " );
        write( l, " phase = " );
        write( l, " " );
        write( l, c );
        writeline( output, l );
        else
        end if;
end process  monitor;
oeh;
```

```
ork.bus_res.all;

ge define is
    type matrix is array( 1 to 3, 1 to 3 ) of real;
    type vector is array( 1 to 3 ) of real;
    type signal_mode is ( v, i );
    type analog_signal is record
    val  : resolve;
    kind : signal_mode;
    end record;
efine;
```

```
age bus_res  is

    type summing is array( integer range<> ) of real;
    function analog( data_array : summing) return  real;
    subtype resolve is analog real;
bus_res;

age body bus_res is
    function analog( data_array : summing ) return real is
       variable i      : integer;
       variable result : real := 0.0;
       begin
          for i in (data_array'low) to (data_array'high) loop
             result := ( result+ data_array(i) );
          end loop;
       return result;
     end analog;

bus_res;
```

```
Example 3,closed loop op apm with 1-zero and 3-pole *
 0 2e8k
 2 0 pwl(0 0v 0.001ns 10v 100ns 10v 100.001ns -10v 200ns -10 200.001ns 10v
       300ns 10v 300.001ns -10v 400ns -10v 400.001ns 10v 500ns 10 500.001ns
         -10v 600ns -10v 600.001ns 10v 700ns 10v 700.001ns -10v 800ns -10v
         800.001ns 10v 900ns 10v 900.001ns -10v 1000ns -10v)
 1 0 1k
 1 8 1 9k
 3 0 laplace {v(2,1)}=
e5*(1.0-s*2.0e-8)/((s*1.0e-8+1.0)*(s*1.0e-7+1.0))}
 0 2e8k
 11 3 0 1e-4
  0 1e4
  4 1u
 5 idealmod
 5 idealmod
 4 idealmod
t 5 6 20
 0 idealmod
 7 0 11 4 1
 7 8 1
 9 idealmod
  8 idealmod
  0 4v
 0 0 -4v
 1 idealmod d n=0.001
t tran v(2) v(8)
  1ns 1000ns
e
```

```
Example 3, closed loop opm  freq response *
 0 2e6k
 0 2e6k
 0 2e6k
 1 0 1k
d 3 1 9k
 3 0 laplace {v(2,1)}=
0e5*(-s*2.0e-8+1.0)/((s*1.0e-8+1.0)*(s*1.0e-7+1.0)*(s*1.0e-2+1.0))}
2 0 ac 1v
nt ac vp(3) vm(3)
dec 10 1kz 10g
be
```